Figure 1 - Examples of Verilog constructs - Prior Art

```
// illustration of Verilog constructs
module verilog_example(out, a, b, reset);
 // Port construct
 output [7:0] out;
 input [7:0] a, b;
 inout reset;

 // Varible construct
 wire internal_reset, q, qn, cp, d, clk;
 reg r1, r2, r3;
 reg [31:0] magic_val;
 reg my_memory [32'hffff, 0];
 integer proc_counter;

 // Parameter construct
 parameter adder_width = 16;

 // Instance constructs
 chip3 arbiter(out, a, b);
 dff1 dff(out[0], qn, cp, d);

 // Gates
 and g1(clk, r1, r2);
 udp3 g2(out[0], qn, cp, d);
 assign (weak0, pull1) #(10, 20, 30) clk = r1 & r2;

 // Scheduled procedural construct
 always @(out[0] or internal_reset)
  begin
   // timing free procedural construct
   r1 = r2;
   magic_val = 0;
   for (proc_counter = 0; proc_counter < adder_width;
    proc_counter = proc_counter + 1)
    begin
     magic_val = magic_val*proc_counte;
    end
   r1 = ^magic_val;
  end

 /* system task construct
 always wait (posedge clk) $display("clk posedge at %t", $time);

 /* user PLI system task construct */
 initial
  begin
   $pli_init_my_memory(my_memory, 1'bx);
  end

 /* specify path and timing check constructs
 specify
  specpararm t0h = 3.0;
  specparam t0l = 5.0;

  (in => out[3]) = (t0h, t0h, t0l, t0l, t0l, t0l);
  $setup(posedge clk, d, 4.33, 2.99);
 endspecify

endmodule
```

Fig. 2

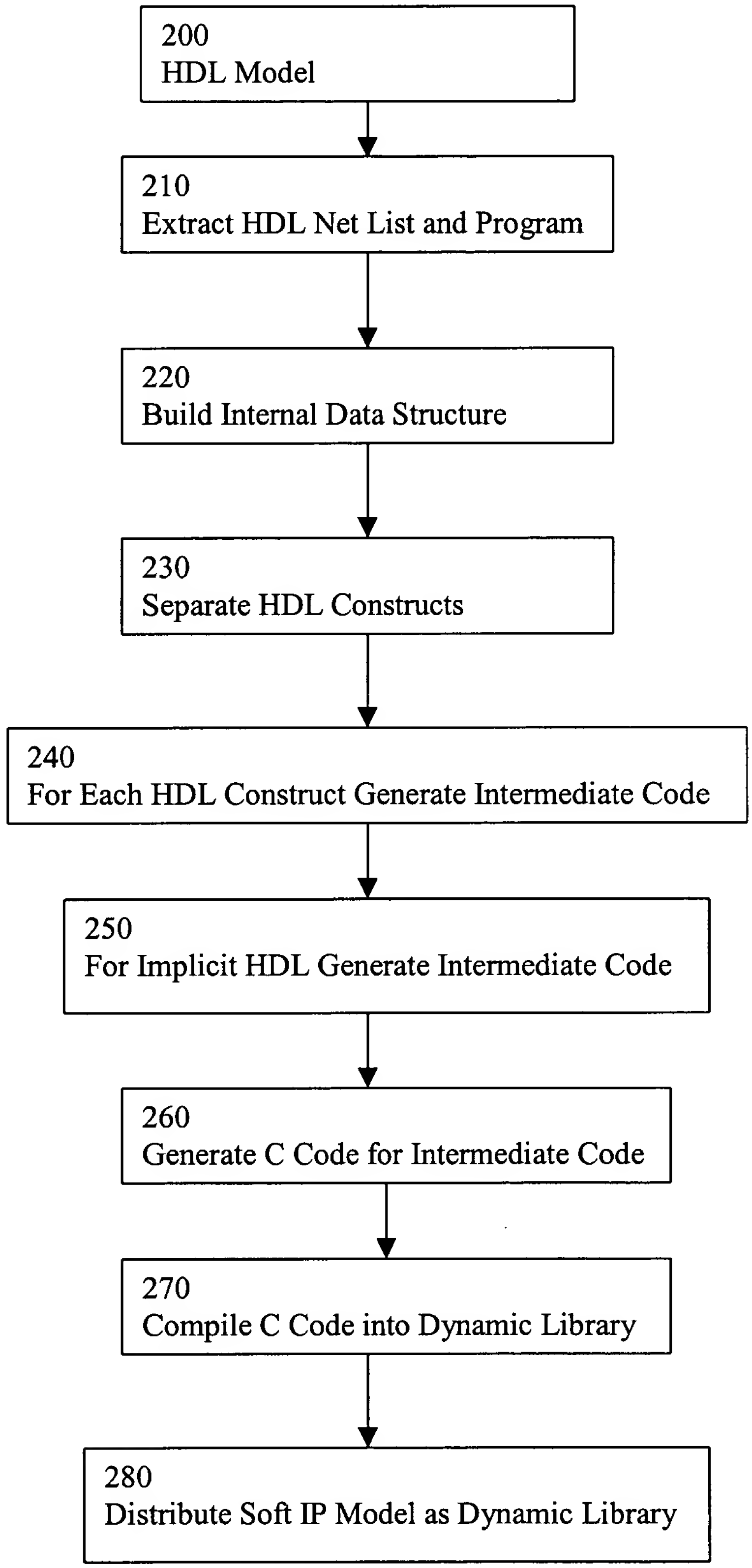

200
HDL Model
210
Extract HDL Net List and Program
220
Build Internal Data Structure
230
Separate HDL Constructs
240
For Each HDL Construct Generate Intermediate Code
250
For Implicit HDL Generate Intermediate Code
260
Generate C Code for Intermediate Code
270
Compile C Code into Dynamic Library
280
Distribute Soft IP Model as Dynamic Library

Fig. 3

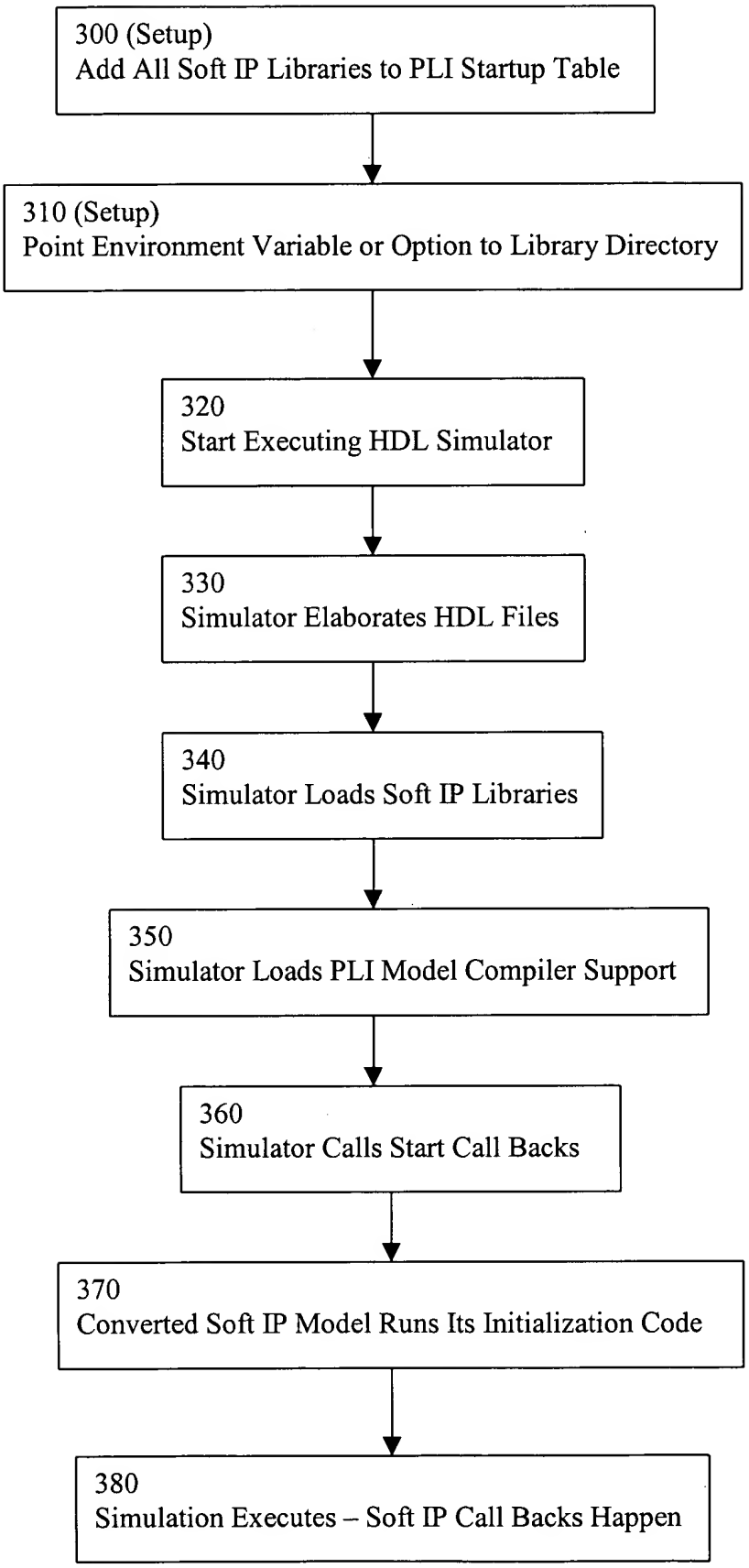

300 (Setup)
Add All Soft IP Libraries to PLI Startup Table
310 (Setup)
Point Environment Variable or Option to Library Directory
320
Start Executing HDL Simulator
330
Simulator Elaborates HDL Files
340
Simulator Loads Soft IP Libraries
350
Simulator Loads PLI Model Compiler Support
360
Simulator Calls Start Call Backs
370
Converted Soft IP Model Runs Its Initialization Code
380
Simulation Executes – Soft IP Call Backs Happen

Fig. 4

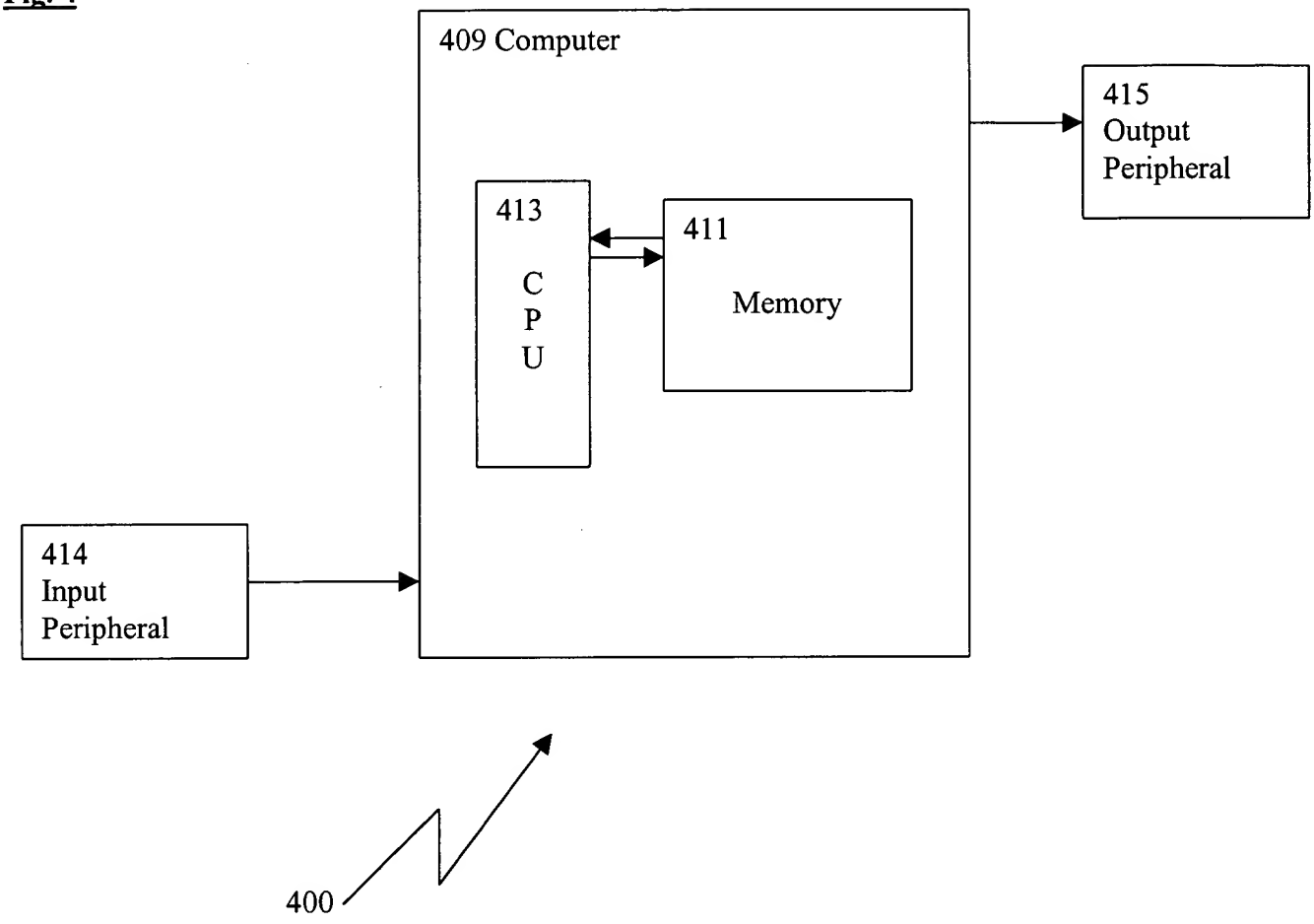

409 Computer
415
Output
Peripheral
413
C
P
U
411
Memory
414
Input
Peripheral
400

Fig. 5

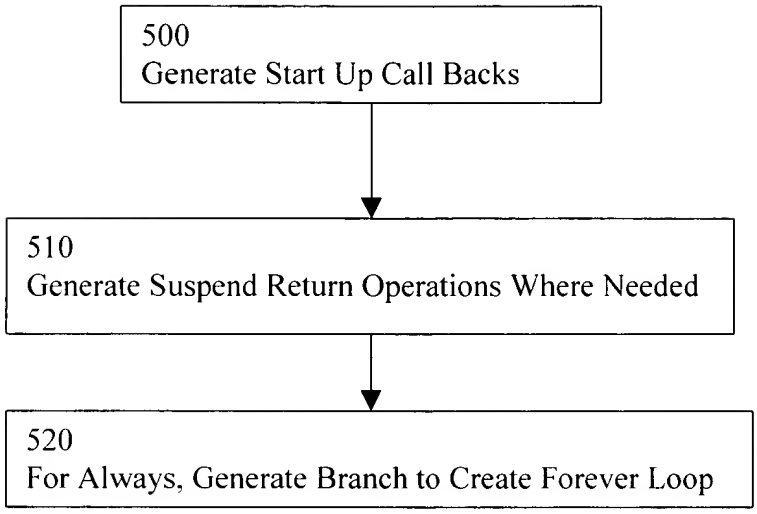

500
Generate Start Up Call Backs
510
Generate Suspend Return Operations Where Needed
520
For Always, Generate Branch to Create Forever Loop

Fig. 6

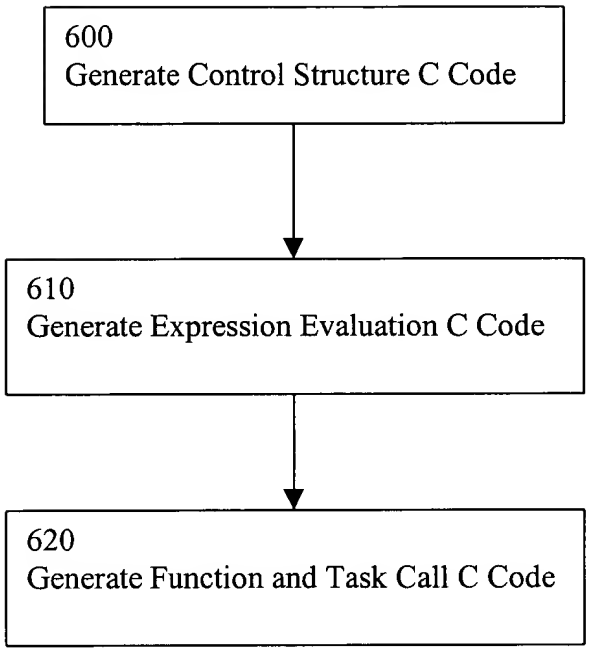

600
Generate Control Structure C Code
610
Generate Expression Evaluation C Code
620
Generate Function and Task Call C Code

Fig. 7

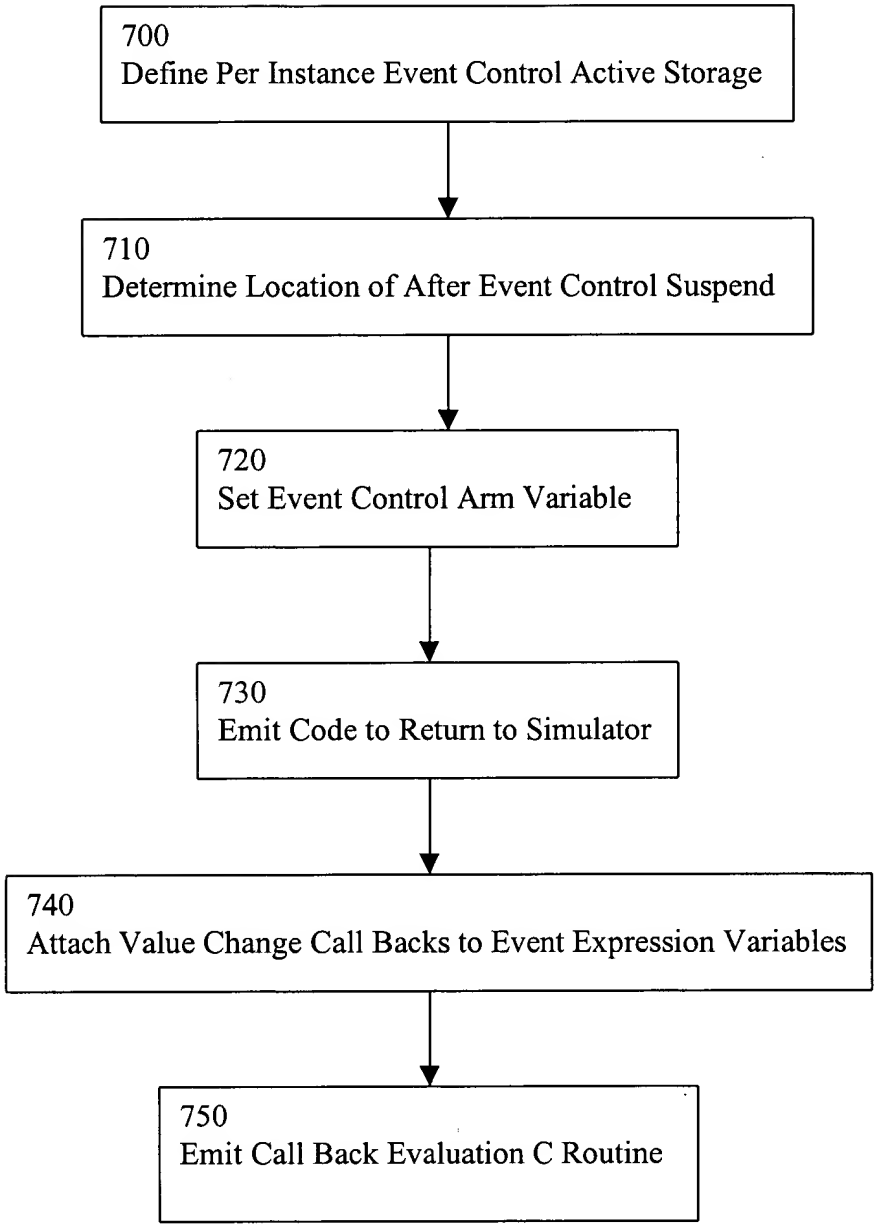

700
Define Per Instance Event Control Active Storage
710
Determine Location of After Event Control Suspend
720
Set Event Control Arm Variable
730
Emit Code to Return to Simulator
740
Attach Value Change Call Backs to Event Expression Variables
750
Emit Call Back Evaluation C Routine

Fig. 8

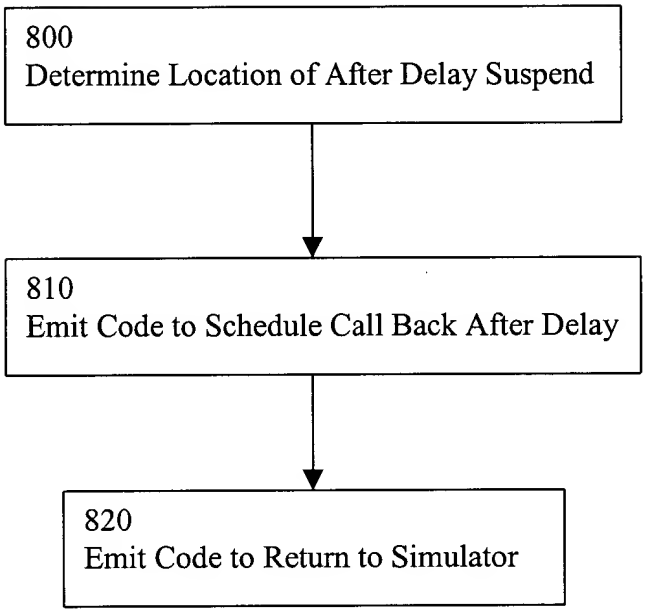

800
Determine Location of After Delay Suspend
810
Emit Code to Schedule Call Back After Delay
820
Emit Code to Return to Simulator

Fig. 9

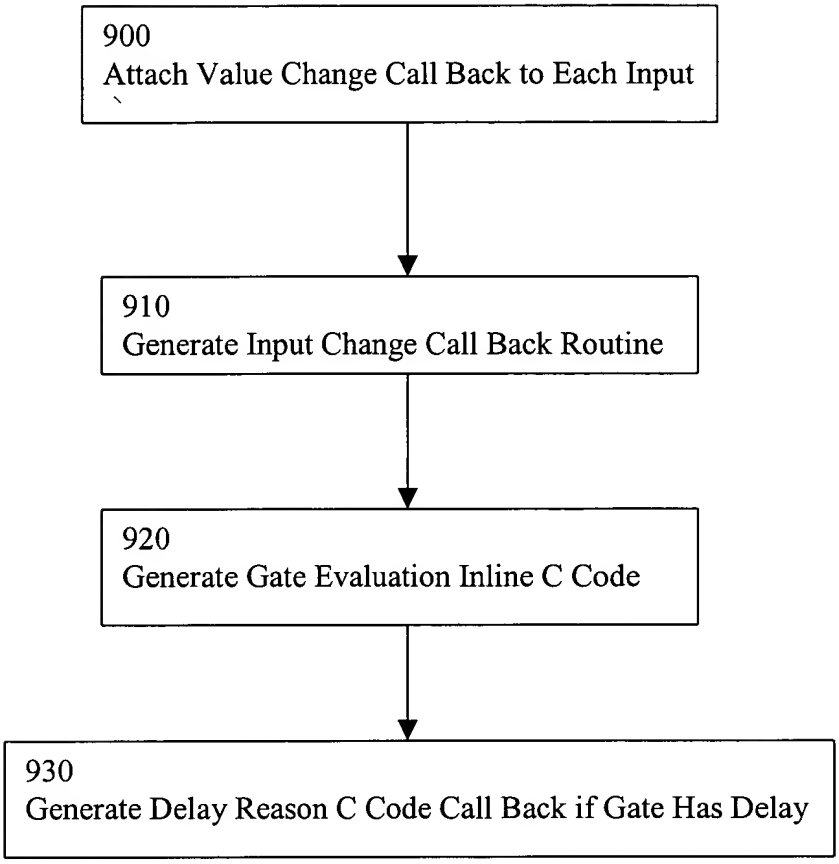
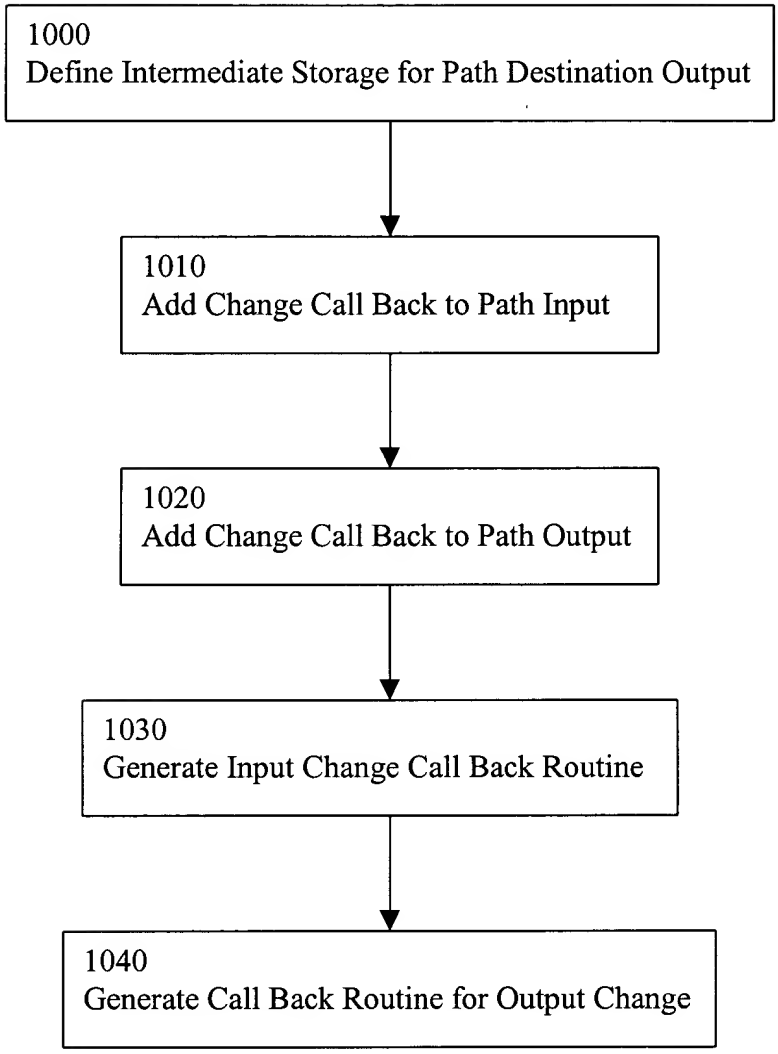

900
Attach Value Change Call Back to Each Input
910
Generate Input Change Call Back Routine
920
Generate Gate Evaluation Inline C Code
930
Generate Delay Reason C Code Call Back if Gate Has Delay

Fig. 10


1000
Define Intermediate Storage for Path Destination Output
1010
Add Change Call Back to Path Input
1020
Add Change Call Back to Path Output
1030
Generate Input Change Call Back Routine
1040
Generate Call Back Routine for Output Change